# 多項式乗算の様々なアルゴリズムの比較 (数式処理における理論と応用の研究)

AUTHOR(S):

紺谷, 拓弥; 野呂, 正行

CITATION:

紺谷, 拓弥 ...[et al]. 多項式乗算の様々なアルゴリズムの比較 (数式処理における理論と応用の研究). 数理解析研究所講究録 1999, 1085: 140-150

ISSUE DATE:

1999-03

URL:

http://hdl.handle.net/2433/62798

数理解析研究所講究録
1085巻 1999年 140-150



# 多項式乗算の様々なアルゴリズムの比較

(株) 富士通研究所 HPC研究センター
紺谷拓弥 (Takuya KONTANI) *

(株) 富士通研究所 HPC研究センター
野呂正行 (Masayuki NORO) †

## 1 はじめに

ネットワークを介したデータ通信の際、データの安全性や機密性を確保するために、様々の方法でデータを暗号化する必要があるが、近年、暗号化の一方式として「楕円暗号」が注目されている。

楕円暗号とは、(任意の体上で) 射影平面上の楕円曲線上の全ての点の集合が群をなすことに注目し、この群における離散対数問題 (「ある群の元、$a$、$b \in < a >$ に対し、$a^n = b$ となる整数 $n$ を求めよ」) の困難さに基づき、暗号化を行うものである。

楕円暗号を生成する際に用いられるのは、有限体上定義された楕円曲線であり、有限体の位数や楕円曲線を定義する方程式の係数 (以下パラメータと言う) の選び方によって、生成される楕円暗号の強度には著しい差が生じる。強い楕円暗号を生成するパラメータを決定する方法としては、様々な方式が提案されてはいるが、未だ決定的な方式を確定するには至っていないのが現状である。

そこで、強い楕円暗号を生成するパラメータを発見するために、計算機を用いた実験が必要となる。整数論で研究されてきた、楕円曲線の性質に関する様々な研究結果を応用すれば、楕円暗号のパラメータを効率よく計算することができる [3]。その際、$f$ を有限体 $GF(p)$ 上の多項式とするときに、$GF(p)$ 上での、

$$x^p \bmod f,$$

なる計算が大量に発生し、計算時間の大きな部分を占める。この計算は、有限体上の多項式の乗算の繰り返しで計算できるので、有限体上での多項式の乗算を高速化することが、楕円暗号のパラメータを効率よく計算するための鍵である、と言うことができる。

*konta@para.flab.fujitsu.co.jp
†noro@para.flab.fujitsu.co.jp

本稿では、数式処理ソフトウエア Risa/Asir にインプリメントした、有限体上の多項式乗算の四種類のアルゴリズムに対して行った性能測定の結果を報告し、アルゴリズムの比較検討を行う。インプリメントされている四種類のアルゴリズムは以下の通りである。

(1) 古典的算法
(2) Karatsuba 法
(3) FFT (Fast Fourier Transform) 法
(4) FFT 法 + Shoup の CRT (Chinese Remainder Theorem) 高速化手法

なお、アルゴリズムの性能の測定は、Sun(hyperSPARC, 150MHz) 及び、Intel(Pentium II, 300MHz) で行った。FFT 法では 32bit 整数を用いて計算を行っているが、比較のために以下の測定も行った。

(3') FFT 法で倍精度浮動小数点変数を用いて計算したもの
(4') FFT 法 + Shoup の CRT 高速化手法で倍精度浮動小数点変数を用いて計算したもの

以下、第 2 章では、Karatsuba 法、FFT 法、 Shoup の CRT 高速化手法の概要に付いて説明する。第 3 章では、性能測定の方法に付いて説明する。第 4 章では、Sun 及び Intel に付いて、測定結果の比較検討を行う。第 5 章では、全体のまとめを行う。

# 2 Karatsuba 法、FFT 法、Shoup の CRT 高速化手法

この章では、数式処理ソフトウエア Risa/Asir にインプリメントされている多項式乗算アルゴリズムである、Karatsuba 法、FFT 法及び、標数が巨大な有限体上の多項式の乗算を行う際に必須である、Chinese Remainder Theorem の高速化手法である、Shoup の手法に付いて説明する。

## 2.1 Karatsuba 法

$f(x)$ 及び $g(x)$ を、$x$ を変数とする $n$ 次多項式とするとき、$f(x)$ と $g(x)$ を $\frac{n}{2}$ 次以下の多項式を用いて

$$\begin{aligned} f(x) &= a(x) \cdot x^{n/2} + b(x), \\ g(x) &= c(x) \cdot x^{n/2} + d(x), \end{aligned}$$

と表現すれば、

$$x(x) \cdot y(x) = a(x) \cdot c(x) \cdot x^n + (a(x) \cdot d(x) + b(x) \cdot c(x)) \cdot x^{n/2} + b(x) \cdot d(x),$$

となり、$n$ 次多項式の乗算が、$n/2$ 次以下の多項式の乗算に帰着できる。同じことを $a(x)\cdot d(x)$ や $b(x)\cdot c(x)$ に対して再帰的に繰り返していけば、多項式の乗算を遂行できる。これは、筆算等を行なうとき通常用いる、古典的算法を再帰的に表現したものであり、一回の reduction につき、乗算 4 回、加算 3 回が必要である。

一方、上式を少し変形して

$$x(x)\cdot y(x) = a(x)\cdot c(x)\cdot x^n+(a(x)\cdot c(x)+b(x)\cdot d(x)+(a(x)-b(x))\cdot(d(x)-c(x)))\cdot x^{n/2}+b(x)\cdot d(x),$$

としてみれば、reduction 一回につき、乗算 3 回、加算 6 回で、計算が行えることがわかる。これが、Karatsuba 法である [4]。

Karatsuba 法の計算量を見てみよう。$T(n)$ は $n$ 次多項式の乗算一回に要するコスト、$C$ は係数の加算 6 回に要するコスト、とすると、

$$\begin{aligned} T(n) &= 3\cdot T(n/2)+C\cdot n, \\ &= 3\cdot(3\cdot T(n/4)+C\cdot n/2)+C\cdot n, \\ &= 3^{\log_2(n)}\cdot T(1)+C\cdot n\cdot\frac{(3/2)^{\log_2(n)}-1}{3/2-1} \\ &= T(1)\cdot n^{\log_2(3)}+2\cdot C\cdot n^{\log_2(3)}-2\cdot C\cdot n, \end{aligned}$$

であるから、計算量が $\mathcal{O}(n^{\log_2(3)})\approx\mathcal{O}(n^{1.58})$ まで削減されることがわかる。但し、加算の回数が 6 回に増えているので、係数 $C$ はかなり大きくなり、次数が非常に低い多項式の乗算に対しては、Karatsuba 法は古典的算法に比較して通常遅くなる。

これを避けるため、Risa には、ある一定の閾値（計算機依存）を設け、reduction の過程で多項式の次数が閾値より小さくなった場合、そこから先は通常のアルゴリズムに切替える方式でインプリメントした。

## 2.2 FFT 法

FFT(Fast Fourier Transform) は、離散 Fourier 変換の高速化アルゴリズムであり、信号処理、画像処理、スペクトル解析等、様々な分野で頻繁に使用されている。

上記のような分野で用いられるのは、浮動小数点演算による複素数体上の FFT であり、基本である Cooley-Tuckey のアルゴリズム以外にも、様々な変形アルゴリズムが考案されている。しかし、FFT は複素数体上のみならず、1 の（適当な）巾根を含む体上でなら、同様に実行可能である。

FFT については、様々な文献で説明されているので [1]、本稿ではその詳細には触れないが、今回、Risa にインプリメントしたのは、東京大学大型計算機センターの村尾裕一先生よりご提供頂いた、基数 2 の Cooley-Tuckey のアルゴリズムによる有限体上の FFT である [5]。

村尾版 FFT は、IEEE 形式の浮動小数点変数の仮数部（24-bit）を用いて、$2^m\times$ 奇数 +1、という形の素数を標数とする有限体上で FFT を実行するように作られている。村尾版 FFT には、上記のような形をした 24bit までの全ての素数と、それらの素数を標数とする有限体の原始元（乗法群の生成元）の表が添付されている。

Risa には、32-bit 整数を用いて計算を行うように改造した FFT もインプリメントした。この場合、使える素数は 29-bit までである。

最後に、FFT の計算量（基数 2 の場合)、及び FFT と畳み込みとの関係について、少しだけ述べる。基数 2 の FFT は、「butterfly 演算」と呼ばれる

$$\begin{aligned} X &= x + W^k \cdot y, \\ Y &= x - W^k \cdot y, \end{aligned}$$

なる演算を、入力データ各々に対し、$\log_2($ 入力データ数 $)$ 回繰り返すことで実行される。但し、$x$、$y$ は入力、$X$、$Y$ は出力、$W^k$ は 1 の（適当な）巾根である。従って、入力データ数を $n$、係数の演算一回のコスト（加算のコストと乗算のコストは同じとする）を $C$ とすれば、FFT の計算量は $\frac{3}{2}\cdot n\cdot log_2(n)\cdot C$ であることがわかる。但し、$W^k$ の計算に要するコストは除いている。また、多項式の乗算とは、その係数を一つの vector と見たとき、vector 同士の畳み込みであり、「時間領域の畳み込みは、周波数領域の積である」という Fourier 変換の定理が、離散の場合にも成立する、ということを思い起こすならば、多項式の乗算が、2 回の FFT と係数毎の積で計算できることがわかる。

## 2.3 Shoup の CRT 高速化手法

現在、高性能暗号の設計に際して求められているのは、その bit 長が 300-bit を越す大素数を標数とする体上での計算であり、そのように大きな整数は、24-bit や 32-bit の変数を用いて一度に処理することは出来ない。そこで、Chinese Remainder Theorem (以下 CRT) が必要となる。

CRT は、整数環のみならず、一般の可換環において成立する定理であるが、今必要なのは整数環 $\mathbf{Z}$ の場合だけであるので、この定理を整数環だけに限った場合の特殊な形で述べるならば、次のようになる。

**定理 1 (Chinese Remainder Theorem)** $a$ を整数、$q_1$、$\cdots$、$q_l$ を素数とし、$P = q_1 \times \cdots \times q_l$ とするとき、

$$\mathrm{mod}(a, P) = \sum_{i=1}^{l} a_i \cdot m_i.$$

但し、$a_i = \mathrm{mod}(a, q_i)$、$m_i = z_i \cdot (P/q_i)$ である。$z_i$ は、$\mathbf{Z}/q_i\mathbf{Z}$ における、$P/q_i$ の乗法に関する逆元である。

この定理は $\mathbf{Z}/P\mathbf{Z} \approx \mathbf{Z}/q_1\mathbf{Z} \oplus \cdots \oplus \mathbf{Z}/q_l\mathbf{Z}$ であることを主張している（$\approx$ は環同型）。つまり、$\mathbf{Z}/P\mathbf{Z}$ の中での $a$ の性質に付いて知りたければ、全ての $i$ に付いて、$\mathbf{Z}/q_i\mathbf{Z}$ の中での $a$ の性質に付いて知ればよい、ということである。

そこで、体の標数 $p$ に対して十分大きくなるように、村尾版 FFT の素数表の素数 $q_1$、$\cdots$、$q_l$ を掛け合わせて整数 $P$ を生成する。そして、各 $q_i$ を法として FFT による多項式の乗算を行い、更に積多項式の係数毎に CRT を実行して、$P$ を法とする係数 $s$ を復元し、最後に $\mathrm{mod}(s,p)$ を計算すれば、最終的な結果を求められる。

では、$P$ としてどれぐらい大きな整数を取ればよいのだろうか。CRT の計算は $(-P/2, P/2)$ で行われることに注意し、扱う多項式の次数の上限を $M$ とすれば、

$$2 \cdot M \cdot p^2 < P,$$

が要求される。$p$ の bit 長が 100 から 300-bit 程度であることを考えれば、$P$ はかなり巨大な整数となることがわかる。

このように巨大な整数を法とした CRT 計算の負荷を削減するために、V.Shoup により一つの手法が考案された [7]。Risa にインプリメントした CRT にもこの手法を採り入れ、高速化した。以下、この Shoup の手法に付いて少し説明する。

積多項式の一つの係数を $a$ とする。$S = \sum_i a_i \cdot m_i$ と書く。$a \in (-P/4, P/4)$ を仮定する。丸め誤差を防ぐため（計算には浮動小数点変数も使用する）、$P$ に 1-bit の余裕を持たせたいからである。

$$\begin{aligned}
a &= \mathrm{mod}(S,P), \\
&= S - P \cdot \lfloor \frac{S}{P} - 0.5 \rfloor, \\
&= \sum_i a_i \cdot m_i - P \cdot \lfloor \sum_i a_i \cdot \frac{z_i}{q_i} - 0.5 \rfloor, \\
t &= \sum_i a_i \cdot \mathrm{mod}(m_i,p) + \mathrm{mod}(-P,p) \cdot \lfloor \sum_i a_i \cdot \frac{z_i}{q_i} - 0.5 \rfloor, \\
\mathrm{mod}(a,p) &= \mathrm{mod}(t,p).
\end{aligned}$$

$\frac{z_i}{q_i}$ の浮動小数点近似、$\mathrm{mod}(-P,p)$、及び $\mathrm{mod}(m_i,p)$ を予め計算しておけば、$l$ 回の (32-bit) 整数と $\log_2(p)$-bit の整数の積、及び $2l$ 回の浮動小数点演算で、CRT が実行できる。これが、Shoup の CRT 高速化手法である。

## 3 測定方法

性能の測定は、Sun(hyperSPARC, 150MHz) 及び、Intel(Pentium II, 300MHz) で行った。FFT 法に対しては、倍精度浮動小数点演算を用いて計算するものと、32-bit 整数演算を用いて計算するものの二通りについて測定を行った。

時間の計測は、Asir の time コマンドを用いて CPU 時間を測定した。多項式の係数の bit 長を 100-bit から 300-bit まで 10-bit ずつ増加させ、各 bit 長に対し、1 次から 200 次までの多項式の乗算に要する時間を測定した。30 回の実行時間の平均値を測定結果とした。

# 4 測定結果の比較検討

この章では、測定の結果得られた、各アルゴリズムの性能について比較検討する。測定の結果は、100、200、及び 300-bit のものをグラフに表示して、本稿の末尾に添付する。

単純に計算量のみを比較するならば、計算性能は、古典的算法、Karatsuba 法、FFT 法、FFT 法 +Shoup の手法、の順によくなるが、Karatsuba 法では、通常アルゴリズムと比較して加算が増加していることや、FFT 法における、1 の巾根の計算や bit reverse に要する負荷を考慮すれば、実際の計算では cross over が発生し、低次数の多項式に対しては古典的算法が、中程度の次数の多項式に対しては Karatsuba 法が、高次数の多項式に対しては FFT 法（+Shoup の手法）が適している、と予想される。この予想は、Sun、Intel の両方に付いて、今回の性能でも裏付けられた。また、Shoup の手法が有効であることも確認できた。以下、Sun、Intel の各々に付いて性能測定の結果を見ていく。

## 4.1 Sun

FFT 法 +Shoup の手法の性能は、どのような次数でも、どのような bit 長でも、FFT 法のみのものより常に上回っていた。大きい bit 長では、Shoup の論文に書かれている通り、2 倍近い性能の向上が見られた。FFT 法において 32bit 整数を使用したものの性能は、浮動小数点変数を使用したものの性能より劣っていたが、係数の bit 長が大きくなるにつれて性能の差が減少していく傾向が見られた。

通常アルゴリズムと Karatsuba 法の閾値、及び Karatsuba 法と FFT 法 +Shoup の手法の閾値は以下のようであった。

| | 通常 →Karatuba | Karatsuba→FFT + Shoup |
|---|---|---|
| 100-bit | 30 | 40 |
| 200-bit | 15 | 40 |
| 300-bit | 15 | 40 |

## 4.2 Intel

FFT 法 +Shoup の手法の性能は、どのような次数でも、どのような bit 長でも、FFT 法のみのものより常に上回っていた。大きい bit 長では、Shoup の論文に書かれている通り、

2倍近い性能の向上が見られた。FFT法において32bit整数を使用したものの性能は、浮動小数点変数を使用したものの性能より勝り、係数のbit長が大きくなるにつれて、更に差が大きくなっていった。Intelの整数演算の性能が優れていることが印象に残った。

通常アルゴリズムとKaratsuba法の閾値、及びKaratsuba法とFFT法+Shoupの手法の閾値は以下のようであった。

| | 通常→Karatuba | Karatsuba→FFT + Shoup |
|---|---|---|
| 100-bit | 30 | 40 |
| 200-bit | 27 | 45 |
| 300-bit | 25 | 50 |

## 5 まとめ

本稿では、第1章で本稿の内容全体に付いての概観を行った。第2章では、Karatsuba法、FFT法、 ShoupのCRT高速化手法に付いて説明した。第3章では、性能測定の方法に付いて説明した。第4章では、Sun及びIntelに付いて、測定結果の比較検討を行った。

Risa/Asirにインプリメントされた、大素数を標数とする有限体上の多項式乗算機能は、当研究センターにおいて暗号研究に活用されている。今回の性能測定で得られた閾値に関する結果も、インプリメントに反映され、いかなる計算に対しても、常に最適なアルゴリズムが呼び出されるようになっている。

今後の課題としては、今回の性能測定の結果を、機械のアーキテクチャに基づいて理論的に解析すること（例えば、bit長が変わるに従って閾値がずれていく現象については、その有意性、原因等、未だ何もわかっていない)、整数版FFT法に使用する素数を32-bitのものまで生成して整数版FFT法の高速化を図ること、FFT法の計算に様々な基数(2の巾、3、5、7等)のアルゴリズムを用いて、FFT法の高速化を図ること(グラフで、FFT法の測定値にギャップが生じているのは、基数2のFFTしかインプリメントされていないからである)、等が挙げられる。

なお、本稿を執筆するに当たっては[2]、[6]を全般的に参照した。

## 謝辞

FFTのソースコードをご提供頂き、多くのご教示を頂いた、東京大学大型計算機センター 村尾裕一先生に謝意を表します。

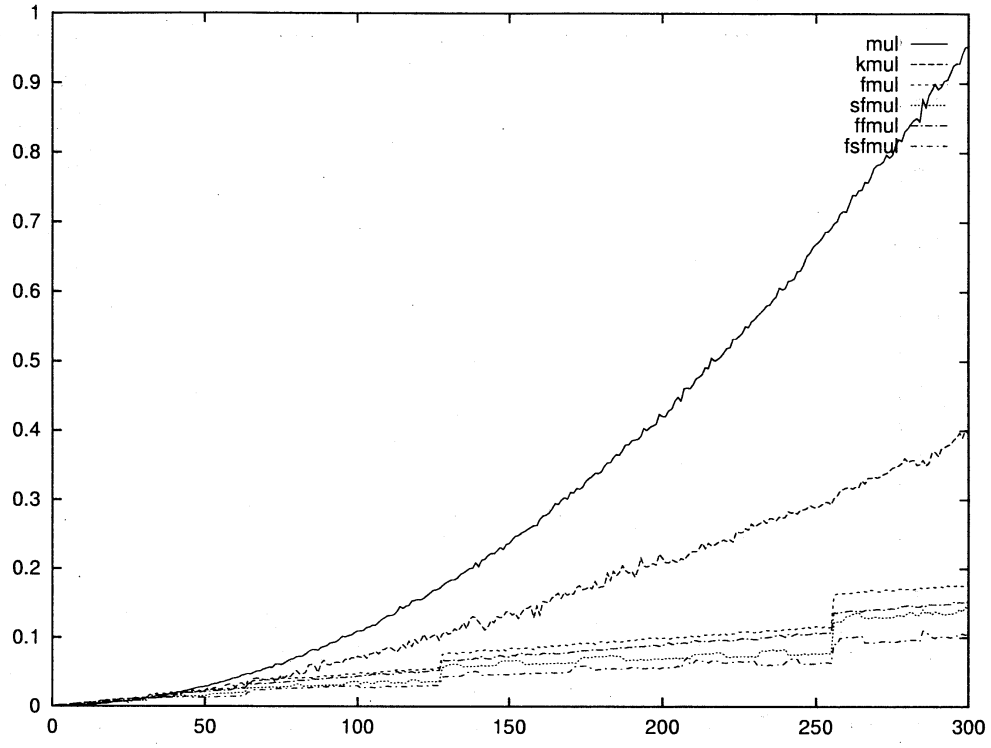


図 1: Sun(100-bit)

## 添付：測定結果のグラフ

Sun 及び Intel の、100、200、300-bit の測定結果のグラフを添付する。グラフの横軸は多項式の次数であり、縦軸は乗算一回に要する CPU 時間 (秒) である。グラフ中の略称は以下の通りである。

mul: 古典的算法
kmul: Karatsuba 法
fmul: FFT 法 (32bit 整数版)
sfmul: FFT 法 + Shoup の CRT 高速化手法 (32bit 整数版)
fmul: FFT 法 (浮動小数点版)
fsfmul: FFT 法 + Shoup の CRT 高速化手法 (浮動小数点版)

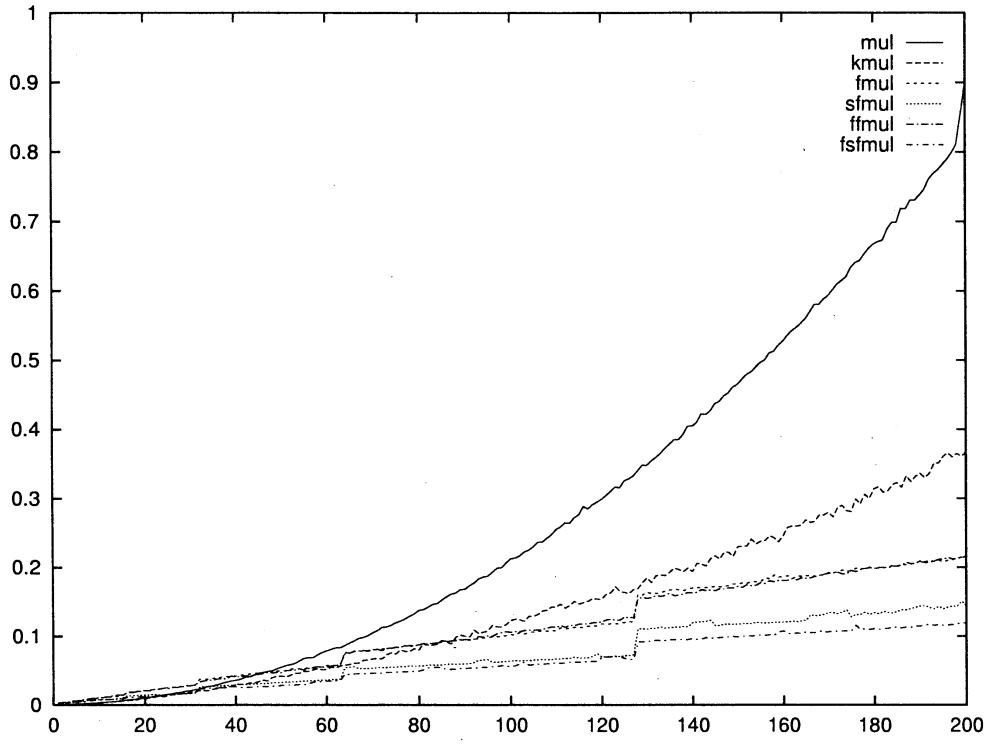


図 2: Sun(200-bit)

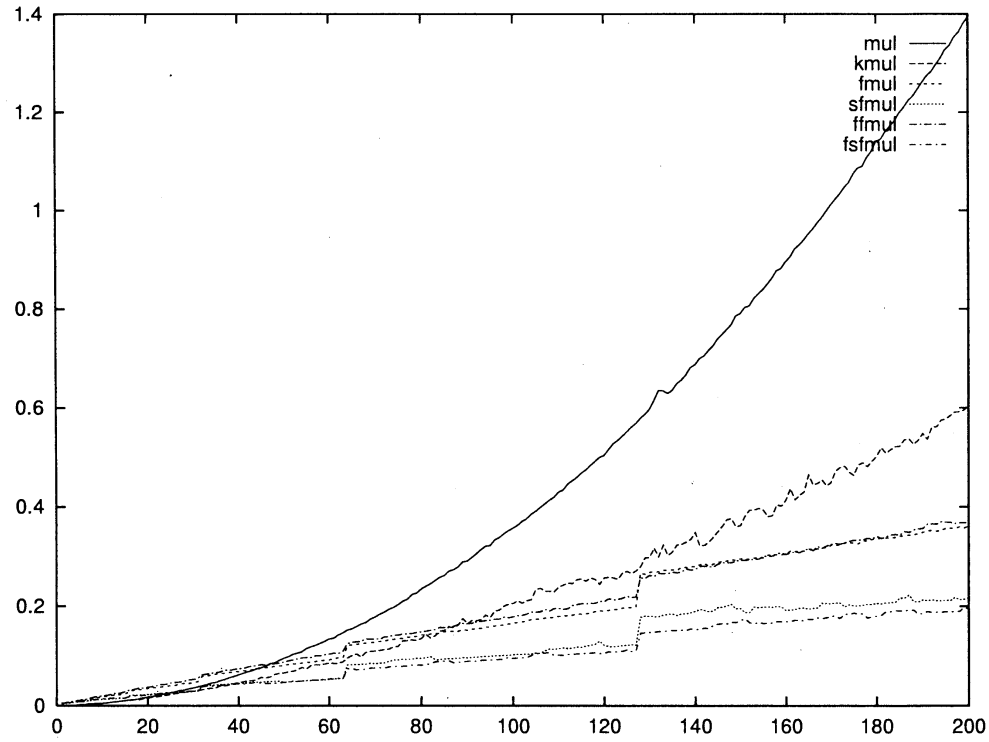


図 3: Sun(300-bit)

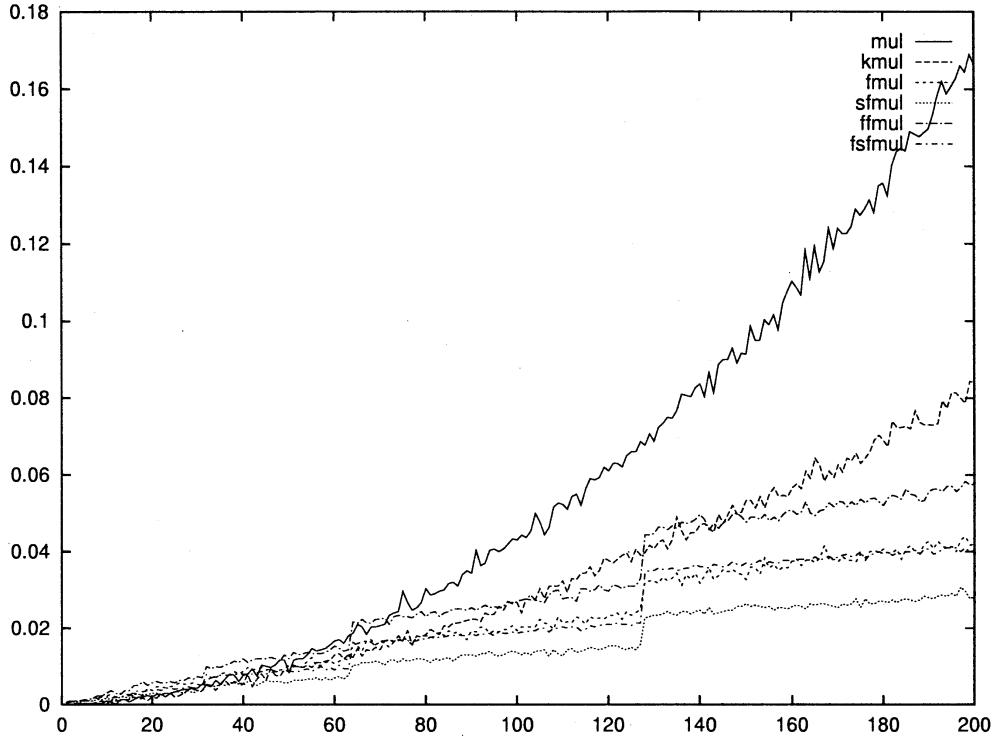


図 4: Intel(100-bit)

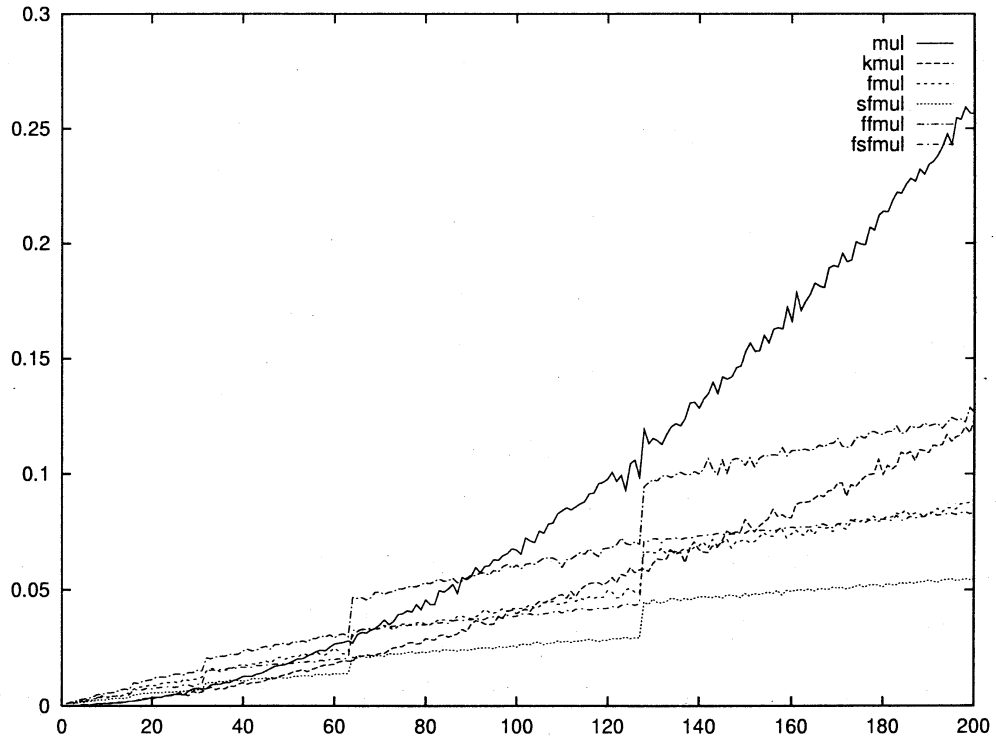


図 5: Intel(200-bit)

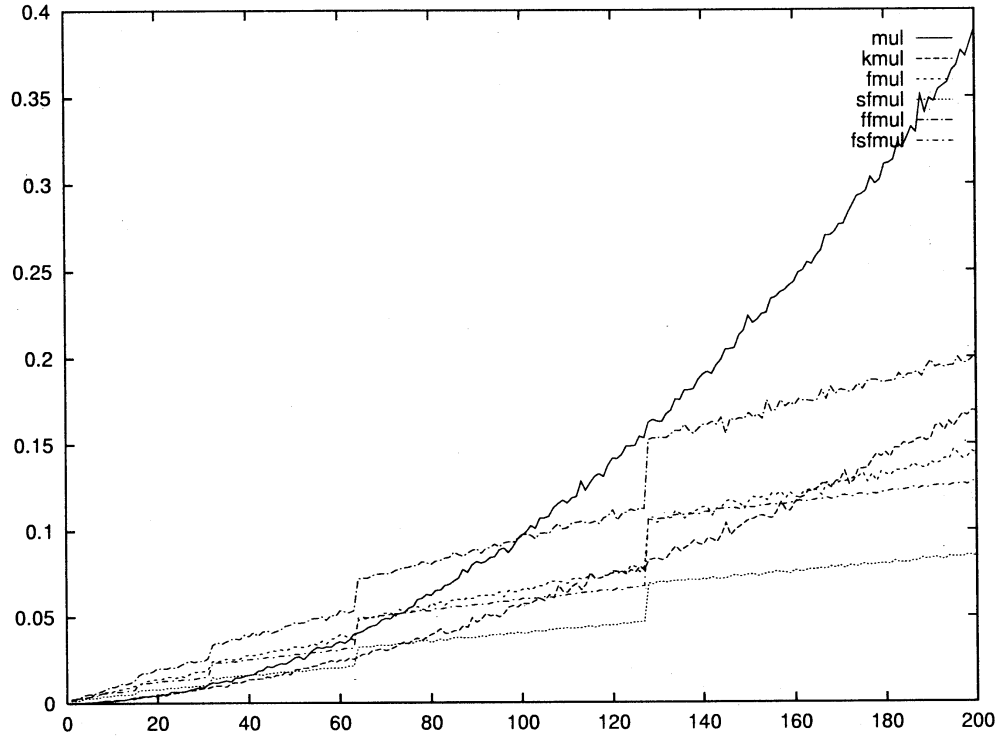


図 6: Intel(300-bit)

# 参 考 文 献

[1] Cooley, J.W. and Tuckey J.W.:An algorithm for the machine calculation of complex Fourier series, *Math. Comp.*, **19**, 1965, 297–301.

[2] Geddes, K., O., Czapor, S., R. and Labahn, G.: Algorithms for computer algebra, Kluwer Academic Publishers, 1992.

[3] Izu, T., Kogure, J., Noro, M. and Yokoyama, K.: Efficient implementation of Schoof's algorithm, in *ASIACRYPT'98*, K.Ohta, D.Pei, Ed., Lecture Notes in Computer Science, **1514**, 1998, 66–79.

[4] Karatsuba, A.:Multiplication of multidigit numbers on automata, *Soviet Physics-Doklady*, **7**, 1963, 595–596.

[5] 村尾裕一:高速多項式乗算アルゴリズムの実現, *J. JSSAC*, **6**, No.1, 1997, 49–50.

[6] 佐々木建昭:情報処理叢書 7 数式処理, オーム社書店, 1981.

[7] Shoup, V.:A new polynomial factorization algorithms and its implementation, *Journal of Symbolic Computation*, **20**, 1995, 363–397.